CLAIMS

What is claimed is:

1. A task management method for determining optimal placement of task components, said method comprising:

a) generating a communication graph representative of a task;

b) identifying independent nets in said communication graph;

c) determining a min cut for each independent net; and

d) placing task components responsive to said min cut determined for each independent net.

2. A task management method as in claim 1, wherein the communication graph generated in step (a) comprises:

task components represented as nodes of said communication graph; and

edges connecting ones of said nodes representing communication between connected nodes.

3. A task management method as in claim 2, after the step (a) of generating a communication graph, further comprising the steps of:

a1) weighting edges, said edges being weighted proportional to communication between connected nodes; and

a2) assigning terminal nodes, task components being placed on said terminal nodes in the task placing step (d).

4. A task management method as in claim 3, wherein the step (b) of identifying independent nets comprises the steps of:

i) selecting a seed node for an independent net;

ii) identifying nodes adjacent to said seed node as perimeter nodes belonging to said independent net, perimeter nodes being an outer perimeter of nodes identified as belonging to said independent net;

iii) identifying nodes adjacent to said perimeter nodes as belonging to said independent net, said identified adjacent nodes being identified as perimeter nodes; and

iv) repeating step (iii) until all perimeter nodes are terminal nodes.

5. A task management method as in claim 4, wherein before the step (i) of selecting a seed node, all nodes not being terminal nodes are marked as unvisited nodes.

6. A task management method as in claim 5, wherein in step (i) the seed node is marked as visited and perimeter nodes are marked as visited in steps (ii) and (iii).

7. A task management method as in claim 6, wherein all nodes marked as visited in steps (i) - (iv) identify an independent net, said method further comprising the steps of:

v) checking said communication graph for unvisited nodes; and,

vi) repeating steps (i) - (v) whenever unvisited nodes remain in said communication graph.

8. A task management method as in claim 3, wherein the step (c) of determining a min cut comprises the steps of:

i) listing all independent nets as subgraphs in a subgraph list;

ii) selecting a subgraph from said subgraph list;

iii) applying a linear complexity method to said subgraph, if said linear complexity method divides said subgraph into two or more smaller independent nets, listing said smaller independent nets in said subgraph list and returning to step (i);

iv) checking whether said subgraph includes two or more smaller independnet nets, if said subgraph includes two or more smaller independent nets, identifying and listing said smaller independent nets in said subgraph list and returning to step (i);

v) applying a higher complexity method to said subgraph, said higher complexity method being more complex than said linear complexity method and, if said higher complexity method divides said subgraph into two or more smaller independent nets, listing said smaller independent nets in said subgraph list and returning to step (i);

vi) selectively collapsing an edge to reduce said subgraph, if collapsing said edge divides said subgraph into two or more smaller independent nets, listing said smaller independent nets in said subgraph list and returning to step (i); and

vii) checking whether said subgraph list is empty.

9. A task management method as in claim 3, wherein each said task component is a unit of the computer program.

10. A task management method as in claim 9, wherein said each computer program unit is an instance of an object in an object oriented program.

11. A task management method as in claim 9, wherein in step (d) computer program units are placed on computers, computer program units being placed on a common computer being combined into a single component.

12. A distributed processing system for determining optimal placement of computer program components on multiple computers, said distributed processing system comprising:

means for generating a communication graph representative of a computer program;

means for identifying independent nets in said communication graph;

means for determining a min cut for each independent net; and

means for placing program components on ones of multiple independent computers responsive to said min cut determined for each independent net;

said computer program being executed by said multiple independent computers.

13. A distributed processing system as in claim 12, wherein the communication graph comprises:

a plurality of nodes, each of said plurality of nodes representing one of said program components; and

a plurality of edges connecting ones of said nodes, each of said edges representing communication between connected nodes.

14. A distributed processing system as in claim 13, further comprising:

weighting means for weighting said edges proportional to communication between connected said nodes and the communication graph further comprises terminal nodes, task components being placed on said terminal nodes.

15. A distributed processing system as in claim 14, wherein the means for identifying independent nets comprises:

means for selecting a seed node amongst the nodes of said communication graph; and means for branching out from said seed node and identifying perimeter nodes adjacent until all perimeter nodes are terminal nodes.

16. A distributed processing system as in claim 15, further comprising means for marking nodes as unvisited nodes or as visited nodes.

17. A distributed processing system as in claim 16, wherein the marking means marks each perimeter node as visited, when all nodes in said communication graph are marked as visited all independent nets have been identified in said communication graph.

18. A distributed processing system as in claim 17, wherein the means for determining a min cut comprises:

means for maintaining a list of all independent nets;

linear complexity reduction means for selectively reducing listed independent nets;

higher complexity reduction means for selectively reducing listed independent nets using a method having higher complexity than used by said linear complexity means;

means for selectively collapsing independent net edges to reduce said listed independent nets;

means for checking whether any reduced independent net includes two or more smaller independent nets; and

means for checking whether said list is empty.

19. A distributed processing system as in claim 18, wherein each said program component is a unit of the computer program.

20. A distributed processing system as in claim 19, wherein said each program unit is an instance of an object in an object oriented program.

21. A computer program product for partitioning a graph, said computer program product comprising a computer usable medium having computer readable program code thereon, said computer readable program code comprising:

computer readable program code means for identifying independent nets in a graph of a plurality of nodes connected together by a plurality of edges each of said edges being between a pair of adjacent said nodes, a plurality of said nodes being terminal nodes;

computer readable program code means for determining a min cut for each independent net; and

computer readable program code means for assigning nodes to ones of said terminal nodes responsive to said min cut determined for each independent net.

22. A computer program product for partitioning a graph as in claim 21, wherein each of said plurality of edges includes a weight, said min cut for said each independent net being a cut of said independent net wherein the sum of weights of edges in said cut is minimum for said independent net.

23. A computer program product for partitioning a graph as in claim 22, wherein the computer readable program code means for identifying independent nets comprises:

computer readable program code means for selecting a seed node; and

computer readable program code means for identifying nodes adjacent to said independent net as perimeter nodes beginning at said seed node.

24. A computer program product for partitioning a graph as in claim 23, wherein the computer readable program code means for identifying independent nets further comprises:

computer readable program code means for marking nodes as visited or unvisited nodes, said seed node and said perimeter nodes being marked as visited.

25. A computer program product for partitioning a graph as in claim 24, wherein the computer readable program code means for identifying independent nets further comprises:

computer readable program code means for checking for unvisited nodes.

26. A computer program product for partitioning a graph as in claim 22, wherein the computer readable program code means for determining a min cut comprises:

computer readable program code means for listing all independent nets as subgraphs in a subgraph list;

computer readable program code means for selecting a subgraph from said subgraph list;

computer readable program code means for applying a linear complexity method to selectively reduce said subgraph;

computer readable program code means for applying a higher complexity method to selectively reduce said subgraph, said higher complexity method being more complex than said linear complexity method;

computer readable program code means for selectively collapsing an edge to reduce said subgraph;

computer readable program code means for checking whether said subgraph includes two or more smaller independnet nets, if said subgraph includes two or more smaller independent nets, identifying and listing said smaller independent nets in said subgraph list; and

computer readable program code means for checking whether said subgraph list is empty.

27. A computer program product for partitioning a graph as in claim 22, wherein each said terminal node represents a computer, each of the remaining ones of said plurality of nodes is a unit of the computer program and each of said edges represents communication between connected computer program units.

28. A computer program product for partitioning a graph as in claim 27, wherein each computer program unit is an instance of an object in an object oriented program.